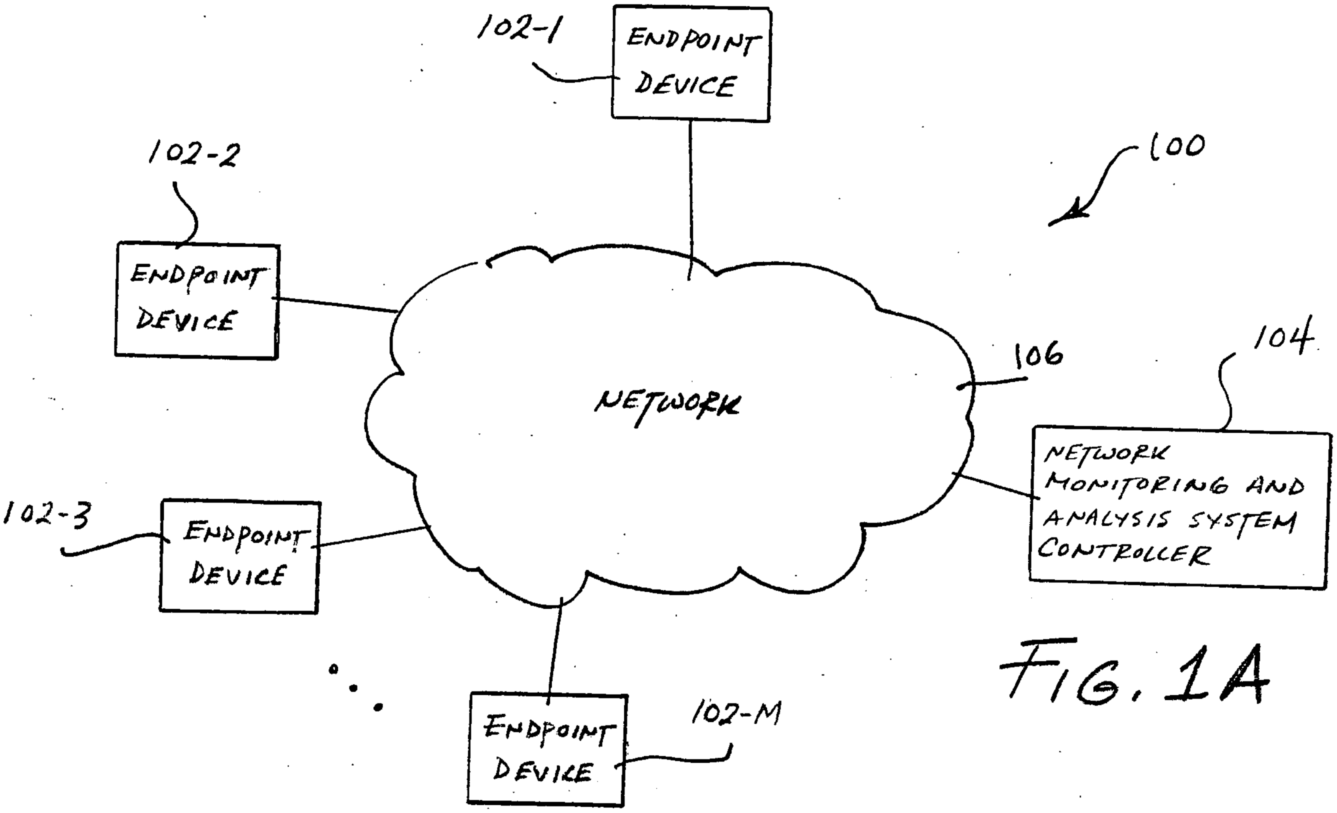
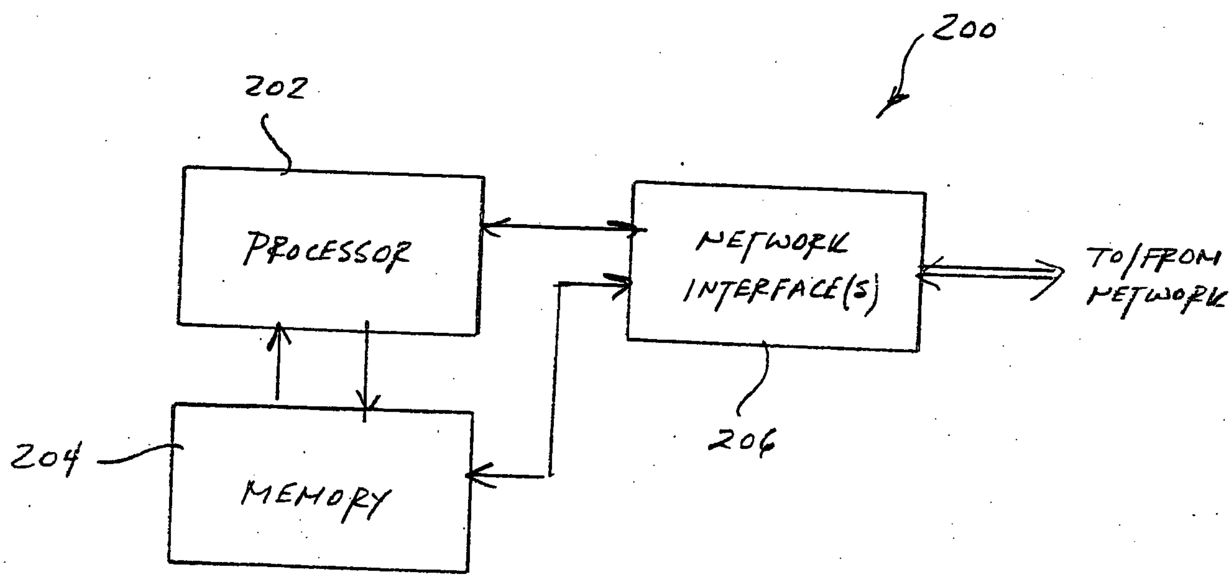


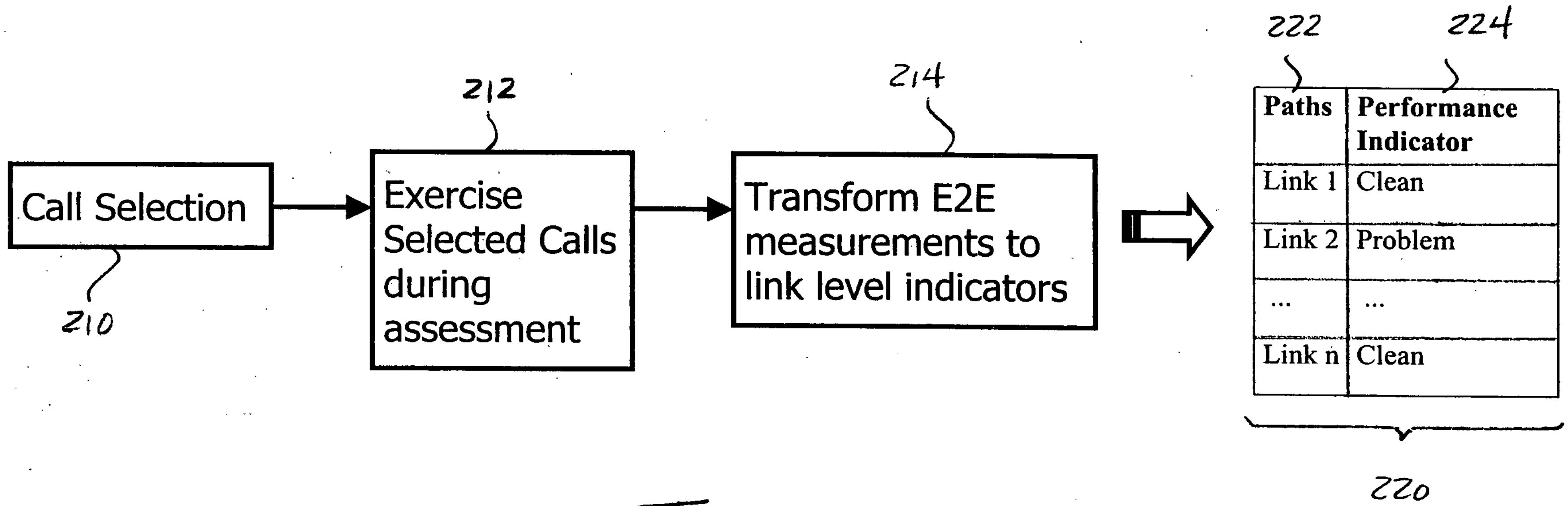
102-1
ENDPOINT DEVICE
102-2
ENDPOINT DEVICE
100
NETWORK
106
104
NETWORK MONITORING AND ANALYSIS SYSTEM CONTROLLER
102-3
ENDPOINT DEVICE
ENDPOINT DEVICE
102-M

FIG. 1A



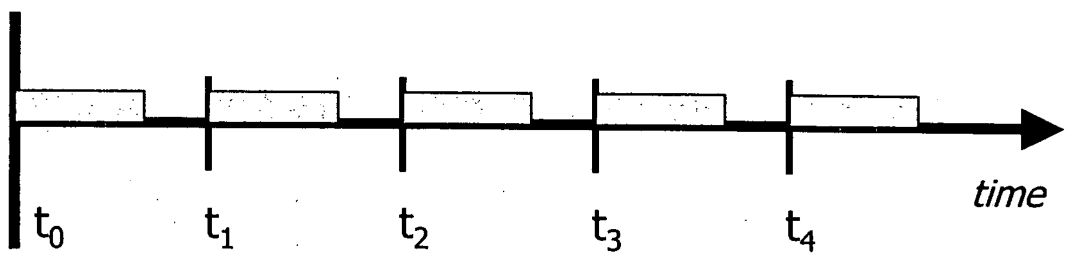
200
202
PROCESSOR
NETWORK INTERFACE(S)
TO/FROM NETWORK
204
MEMORY
206

FIG. 1B



Call Selection
210
Exercise Selected Calls during assessment
212
Transform E2E measurements to link level indicators
214
222
224
Paths
Performance Indicator
Link 1
Clean
Link 2
Problem
...
...
Link n
Clean
220

FIG. 2A



t_0
t_1
t_2
t_3
t_4
time

FIG. 2B

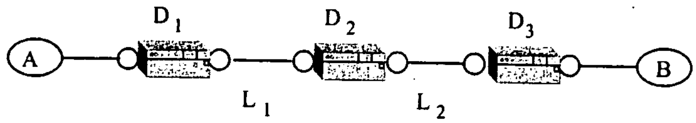


FIG. 3

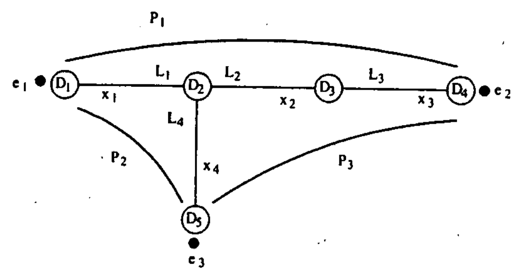


FIG. 4

	L_1	L_2	L_3	L_4
P_1	1	1	1	0
P_2	1	0	0	1

Flow matrix 1

	L_1	L_2	L_3	L_4
P_1	1	1	1	0
P_2	1	0	0	1
P_3	0	1	1	1

Flow matrix2

FIG. 5

Equations with Flow matrix 1

$$\begin{aligned} x_1 + x_2 + x_3 &= y_1 \\ x_1 + x_4 &= y_2 \end{aligned}$$

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Equations with Flow matrix 2

$$\begin{aligned} x_1 + x_2 + x_3 &= y_1 \\ x_1 + x_4 &= y_2 \\ x_2 + x_3 + x_4 &= y_3 \end{aligned}$$

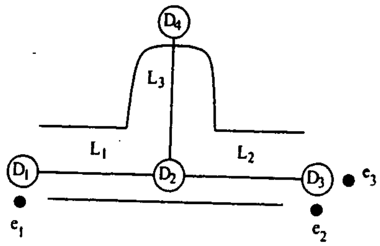
$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

FIG. 6

Generate_Pipes($G = (D, L)$):Network Topology Graph, E: Set of Leaves)

I: Set of pipes in G wrt E
$I \leftarrow \emptyset$
Compute P for G wrt E
Let M be the complete flow matrix for G and P
// Group links with the same column vector into disjoint sets
Let k be the number of distinct column vectors in M
Form a set $\mathbf{S} = \{S_0, S_1, ..., S_k\}$ where :
  each S_i, $0 < i \leq k$ contains links in L with the i^{th} distinct column vector in M
// Ensure that links in each element of **S** *form a path in G*
for i=1 to $|\mathbf{S}|$
  if links in S_i are consecutive and form a path
    then merge S_i into path p, $I \leftarrow I \cup \{p\}$ *// add the path formed by the links in S_i as a pipe*
    else $I \leftarrow I \cup S_i$ *// add each link as a pipe by itself*
return I

FIG. 7



$$
\begin{array}{c|ccc}
 & L_1 & L_2 & L_3 \\
\hline
e_1 - e_2 & 1 & 1 & 0 \\
e_1 - e_3 & 1 & 1 & 2 \\
e_2 - e_3 & 0 & 0 & 0
\end{array}
$$

FIG. 8

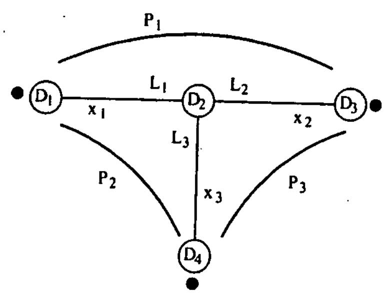


FIG. 9

Flow matrix 1

$$
\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}
$$

Flow matrix 2

$$
\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}
$$

FIG. 10

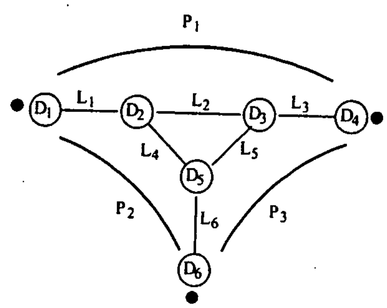


FIG. 11

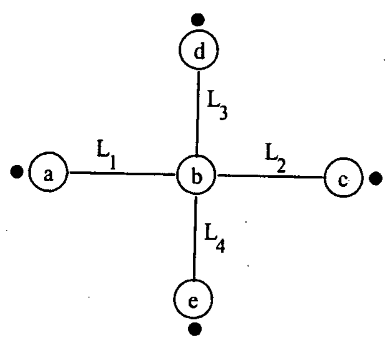


FIG. 12

	L_1	L_2	L_3	L_4
$L_1.L_4$	1	0	0	1
$L_4.L_2$	0	1	0	1
$L_2.L_3$	0	1	1	0
$L_3.L_1$	1	0	1	0

$\{L_1.L_4, L_4.L_2, L_2.L_3, L_3.L_1\}$

	L_1	L_2	L_3	L_4
$L_1.L_2$	1	1	0	0
$L_1.L_3$	1	0	1	0
$L_1.L_4$	1	0	0	1
$L_2.L_3$	0	1	1	0

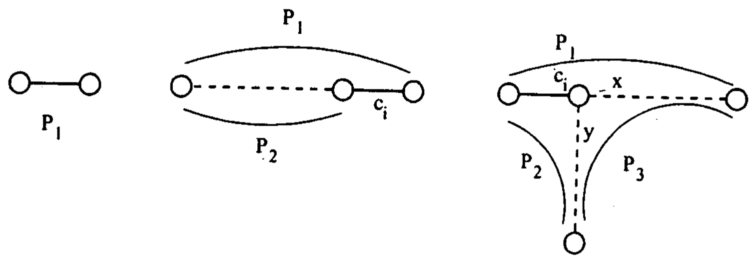
$\{L_1, L_2, L_3, L_4\}$

FIG. 13

Select_Matrix($G' = (D', I)$):Reduced Network Topology Graph, E: Set of Leaves)

W: Set of worms in G' wrt E, $W \leftarrow \emptyset$
R: Set of paths, $R \leftarrow \emptyset$
Compute P' for G' wrt E
$open \leftarrow P'$
while $open \neq \emptyset$
    select p from $open$
    for each pipe c_i on $p = c_1.c_2.\ldots.c_{length(p)}$
        $\boxed{\text{if } \exists S \subset open \text{ such that } S \text{ makes } c_i \text{ estimable}}$ $\leftarrow$ B1
            Compute S' which has the original value of each path in S
            $R \leftarrow R \cup S'$
            $W \leftarrow W \cup \{c_i\}$
            update $open$ and W such that $\forall p' \in open$
                p' does not contain any estimable path in W *// c_i is removed from paths in open*
        else
            $c_{i+1} \leftarrow c_i.c_{i+1}$
    $open \leftarrow open \setminus \{p\}$
return W, R

FIG. 14



$c_i = P_1$

$P_2 \bullet c_i = P_1$

$c_i \bullet x = P_1$
$c_i \bullet y = P_2$
$x \bullet y = P_3$

Pattern 1

Pattern 2

Pattern 3

FIG. 15

Compute_EstPaths($G' = (D', I)$:Reduced Network Topology Graph, E: Set of Leaves, P'_{t_i}: End-to-end paths at time t_i)

```
M: A Minimal set of estimable paths for G' wrt E, M ← ∅
open ← P'_{t_i}
while open ≠ ∅
    while open not converged
        select p from open
        for each pipe c_i on p = c_1.c_2. . . . c_{length(p)}
            if ∃S ⊂ open such that S makes c_i estimable
                M ← M ∪ {c_i}
                update open and M such that ∀p' ∈ open
                    p' does not contain any estimable path in M and        // c_i is removed from paths in open
                    open ← open \ {p}
        else
            abort processing of p
    if open ≠ ∅
        select shortest p in open
        open ← open \ {p}
        M ← M ∪ {p}
return M
```

FIG. 16